Model-based Cybersecurity Engineering for Connected and Automated Vehicles: The FLOURISH Project

David Robles-Ramirez[1], Theo Tryfonas[2,*], Ponciano J. Escamilla-Ambrosio[1], Tesleem Fagade[2], Kalliopi Anastasopoulou[3], Andrea Tassi[2], Robert Piechocki[2]

[1] Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico City, Mexico
[2] University of Bristol, Bristol, U.K.
[3] R&D Dept, 7th Healthcare Region, Ministry of Health, Greece
*contact author: theo.tryfonas@bristol.ac.uk

Abstract

Connected and automated vehicles (CAVs) represent a challenge for future transportation systems as they generate a massive amount of data which may also include security threats and vulnerabilities for users. In this paper, we adapt a model-based systems engineering (MBSE) approach called the Internet of Things Security Modelling (IoTsecM) to address security challenges and system-level security critical issues in the domain of CAVs. Not only are connected and automated vehicles considered, but also their interactions with other assets such as roadside infrastructure, sensors and traffic lights. The application is based on a project which identified innovative solutions related to connectivity, data analytics and safe design for CAVs in the UK. The objective of introducing IoTsecM into the project context was to provide an MBSE method to develop a systems architecture where the security mechanisms and controls are identified and modelled during the requirements stage in order to facilitate secure, trustworthy and private CAV technology development by design.

1 Introduction

The Internet of Things (IoT) uses the internet to provide information transfer services, analytics, applications, and communications, meaning that more objects (e.g. cameras, wearables, environmental sensors, appliances etc.) are 'connected', generating massive amounts of data. The collection, integration, processing and analytics of that data enables the realization of smart environments, infrastructures and services for enhancing quality of life (Porter & Heppelmann, 2014). Connected and automated vehicles (CAVs) in particular are recognized as a smart-connected asset in

DOI:https://doi.org/10.6028/NIST.AMS.100-29#page=149

future transportation systems, with the IoT playing a key role in the connectivity and access of their multiple component systems (McCarthy et al., 2016). The network infrastructure collects data from the environment leveraging sensing capabilities and interacts with the physical world by performing actuation and command-and-control over other things (Rigazzi et al., 2017).

However, as the number of entities connected to the internet grows, the security attacks surface grows too. IoT systems pose new challenges that were not present in the traditional internet. Security requirements of IoT systems are often considered as an after-thought, even when the information handled by those systems is sensitive (Conti et al., 2018). In particular, the risk of cyber-attacks directed to CAV applications can compromise the availability and integrity of passenger information, crippling mobility and even threaten passengers' safety, if decisions are made based on invalid information (Baig et al., 2017; Woo et al., 2015). The vast amount of data and the considerable level of risk associated with CAV operations makes them a challenging area for security analysis. Much work is required to develop systems and methodologies for handling operational security requirements, while maintaining a high level of privacy for the individual users (FLOURISH, 2019).

In this paper, we present a method for comprehensive analysis of the threats and risks associated with connected and automated vehicles and their potential impact on the transportation system. We adopt a model-based systems engineering (MBSE) approach called the Internet of Things Security Modelling (*IoTsecM*, Robles-Ramirez et al., 2017) and combine it with attack tree security analysis methodology to address security threats, vulnerabilities and system-level security critical issues.

The following section 2 reviews the context of connected and automated vehicles, security threats and vulnerabilities and the role of IoT in CAVs and the transportation system. Section 3 provides an illustration of the approach's use in the context of a real-world project aiming to design secure CAVs in the UK and addressing traffic safety, congestion, and user behavior. The final section concludes our work and explores further opportunities for CAV security research in the MBSE space.

2 Background Work

2.1 Technical Components Architecture and Threats

Driverless vehicles generate large sets of data in the interconnected world. CAVs hold GPS and integrated infotainment systems, which can link to smartphones and to cloud computing (Samie et al., 2016). Vehicles with GPS and traffic light sensors produce vehicle location data, while a vehicle follows a specific direction for the user and their travel. Many vehicles today use the united diagnostic services protocol (UDS) for diagnostics and interaction with the onboard diagnostics unit (OBD). The OBD communicates with Bluetooth and smartphone communication systems, which use the vehicle network to exchange data and contact the network control center (Woo et al., 2015). CAVs have also essential capabilities of recording and storing data. The event data recorder (EDR) records data such as speed variations, which can be associated with accident events (Mansor et al., 2016). For example, the electronic control units (ECUs) can record speed data or data related to a journey distance. Many driverless vehicles have also integrated EDRs which can even record personal data (e.g. the number of occupants in the vehicle, if they are identifiable). A lot of data collected from CAVs can be personal (e.g. geolocation, MAC address, user ID etc.) or depending on application, even more sensitive information (e.g. indicators of the physical or mental health of an occupant). They may also carry commercially sensitive data (e.g. intellectual property) or non-sensitive data (e.g. traffic congestion or speed).

Data produced by CAVs may pose security challenges for the vehicle and its users. Data protection, safeguarding from physical damages, and any other threat and vulnerability is of high interest for a secure and trusted operation. These security concerns may stem from the following

sources: physical (e.g. side channel attacks to crack information), interception (such as man-in-the-middle attacks), abuse (such as unauthorized access to the vehicle), malicious code (generic malicious code affecting the integrated infotainment system), data leaks (e.g. when the vehicle changes owner) etc. (Samie et al., 2016). All the above security threats are inextricably linked with the IoT, as it is the means by which the components identified above communicate, exchange data, decide, take actions and provide services, thus shaping the connected transportation ecosystem.

In general, some of the key challenges for IoT in relation to security that also apply to CAVs include: a) naming and identity management, b) interoperability and standardization, c) information privacy, d) objects safety and security, e) data confidentiality and encryption, f) network security, g) spectrum allocation etc. (Khan et al., 2016). In the world of CAVs, IoT specifically applies to connecting sensors and vehicles to networks (McCarthy et al., 2016). Khan et al. (2016) refer to IoT sensors such as traffic congestion sensors that collect data and store them in cloud servers. Here some security threats are in particular: a) confidentiality and integrity; b) eavesdropping; c) data loss; d) availability compromise and e) remote exploitation.

Therefore, in an interconnected environment where IoT is integrated with vehicles, vulnerabilities of embedded systems that can lead to cyber-attacks is a real concern with economic and physical implications. Addressing those cybersecurity issues is challenging because those systems are complex and with limited computational power (Samie et al., 2016). To ensure security, a comprehensive design and operation process is required. Model-based systems engineering with the extension of UMLsec is capable of incorporating a security viewpoint to the system (e.g. Jürjens et al., 2008; Oates et al., 2013; Apvrille & Roudier, 2016). In this paper we propose an approach which can be used to identify threats and vulnerabilities for CAVs early and help a designer to integrate security controls to the overall system.

2.2 The FLOURISH Project

CAVs will have significant impact in many aspects of our lives from a technological, social and economic perspective when becoming a reality. The United Kingdom aims to become one of the most considerable actors in the world of driverless vehicles, a goal with a prerequisite for a secure cyber environment (McCarthy et al., 2016). Integrity and clarity on sharing data, as well as the development of cybersecurity standards, will be fundamental to support the development of CAV technology. In this national context, FLOURISH (n.d.) was a multi-sector collaboration, helping to accelerate and promote the successful implementation of CAVs in the UK, by establishing services and capabilities that link user needs and system requirements. The project aimed to address cybersecurity threats and privacy issues by design, as well as explore user acceptance of CAVs. Assessing cyber risks is the key component for the protection of CAVs, but they are not an isolated system and so this is not a trivial task. CAVs operate within a more extensive network and a complex infrastructure. Also, exploring user-acceptance requires transparency on how CAVs use user data and how they are protected from cyber-attacks (FLOURISH, 2019).

The main security threats that the project was concerned with were a) loss of control over the system as the result of cyber-attacks; b) damage or loss of technology assets (e.g. loss of data or damage caused by a third party); c) any abuse such as denial of service attack or unauthorized access to systems; d) information leakage or sharing, inadequate design and planning or lack of adoption of standards; e) failures or malfunctions (e.g. software bugs); f) information interceptions or network reconnaissance. Not only automated vehicles are considered, but also the interaction of these with other assets such as city infrastructure, sensors and traffic lights.

2.3 The IoTsecM Security Modelling Extension

The main objective of threat modelling is to identify system vulnerabilities which could be exploited by a motivated attacker and understand how they could be exploited, if countermeasures are not implemented (Shostack, 2014). Once possible attacks over the system are identified, the developer is able to specify the required protection or countermeasures against those attacks. Threat modelling can be achieved in different ways, as there is no unique methodology. Various sources provide methods and tools for it, e.g. Microsoft proposes an approach which uses multiple steps to determine the severity of threats, the secure development lifecycle (SDL) (Shostack, 2014).

For the requirement of security-by-design posed by the FLOURISH project, we explored the use of an MBSE approach to IoT security modelling that could serve the nature of the highly interconnected CAV components. We decided to adopt the IoTsecM approach (Robles-Ramirez et al., 2017) that employs a UML/SysML extension nomenclature to consider security requirements along the analysis stage of a well-defined development life cycle, such as the waterfall model. Some of its nomenclature components have been previously proposed in the IoT-A proposalas security modules (M. Unis et al., 2013). However, they were not proposed as a UML/SysML extension. As such, IoTsecM provides a set of well-defined elements which abstract the security features of IoT systems and allows for them to be embedded in UML/SysML diagrams.

Therefore, it is a notational representation which integrates security elements in a nomenclature and encapsulated in stereotypes. Such stereotypes are presented in Table I. Each one of the elements encapsulates an IoT security service, and has a short representation (nomenclature elements), the corresponding UML extension mechanism and the metaclasses extended by the element. These elements are used inside the extended UML/SysML diagrams because they are high level abstraction security requirements and they encapsulate the traditional 'CIA' security goals (i.e. confidentiality-integrity-availability).

The nomenclature can be applied as use cases, if the security requirements warrant it. In a class diagram, some nomenclature elements can be modelled as classes, and, in fact, according to the IoT security requirements analysis performed, some of these elements are: N, Z, C and D (as identified in Table I). The TP, SS and SC elements are constraints represented as [TP], [SS] and [SC], and these three elements are used mainly in the use case diagrams and in UML behavior diagrams. For example, the D (decryption requirement) and N (authentication) elements can also be modelled as use cases. This allows a more agile design process for security requirements, because even though developers are not necessarily involved in security analysis, they can still recognize these elements.

Element	**Name**	**Extension mechanism**	**Base meta-class(es)**
N	Authentication	Stereotype	Class, use case, component, block, activity and state
Z	Authorization	Stereotype	Class, activity, component, block, state and use case
C	Cipher	Stereotype	Use case, component, block, class
D	Decipher	Stereotype	Use case, class and component
SS	Secure Storage	Stereotype	Link, property, association, communication path and constraint
SC	Secure communication	Stereotype	Constraint, communication path and link
TP	Tamper protection	Stereotype	Constraint and property

Table I: IoTsecM nomenclature

A sequence diagram depicts objects interactions chronologically, where the classes which apply the nomenclature appear in the diagram as objects. A potential attack will be ultimately modelled as a sequence diagram, having first been identified and understood in this analysis via the use of attack trees. Attack trees are one way to model attacker behavior against the system assets (Apvrille & Roudier, 2016). Normally, an attack is grouped in a sequence of sub-attacks or other activities that are individually focused on obtaining an immediate target. Attack trees let us model these sub-attacks and the steps that need to be followed to obtain the target. This attack representation helps to conceptualize, visualize, and communicate a better understanding of the vulnerabilities that could be exploited and then is can be easily translated to a sequence diagram embedded in the overall model.

3 Model-based Cybersecurity Engineering for CAVs

Many threats could undermine the FLOURISH architecture, given that CAVs are in constant interaction with their environment. They could become subject of interest to persistent and motivated attackers. The objective of introducing IoTsecM into FLOURISH was to provide an application architecture where the security mechanisms and controls are specified and modelled in order to enable secure, trustworthy and private technology development within CAVs and across the whole infrastructure (Rigazzi et al., 2017). The IoTsecM extensions provide a notation and semantics which model and depict the security requirements in the system architecture model. In this work we adopt the threat modelling approach originally described in (Robles-Ramirez et al., 2017). The approach was customized and extended in order to add the countermeasures modelling. The process followed is summarized into the following steps: 1) Identify the assets, 2) Create an IoT system architecture overview, 3) Decompose the IoT system, 4) Identify threats, 5) Document threats, 6) Propose counter-measures for each threat, 7) Propose a system architecture with security countermeasures.

Asset	**Description**
LIDAR	It is a sensor located in strategic places and it creates BBR data (data monitored from other cars)
CAVs	The connected and automated vehicles
RSU	The roadside unit
Carer	Is the person dedicated to activating the point in order to establish a special zone which is a restriction zone for special requirements such as slow traffic
Instructions data	It is the result data delivered by the control room and sent to CAVs through the RSU
On board sensors	They are sensors located within the CAV mainly to monitor passengers
Vehicle level AI unit	Artificial intelligence unit within the CAVs
Autonomous control system	The subsystem which carries out the CAVs control

Table II: Flourish architecture components (assets)

Assets are system components which are of interest to an attacker; they can be hardware, software, physical entities or even humans. Assets identification allows an understanding of what must be protected to mitigate the impact of threat. In the context of this study, the assets were obtained by

analyzing the scenarios provided by the FLOURISH team, who described each scenario as general system use-cases. The FLOURISH architecture involves automated vehicles communicating with each other and with human-driven vehicles (HDV) and to roadside units (RSU). This communication is referred to as V2X (vehicle to everything). The system architecture overview consists mainly of CAVs travelling around the city with potential passengers riding. The RSU is the communications hub that is strategically located to communicate key commands to the CAVs and link them to diverse processing centers. Therefore, the system architecture comprises three assets categories: a) CAV, b) RSU, and c) Processing nodes. Due to space constraints, only a few of the assets identified from the scenarios are shown in Table II.

Once the assets were identified, the next step involved threat modelling as described earlier, to create a system architecture overview. The original system architecture is depicted as a UML class diagram in Figure 1. It contains the assets identified and their interconnections. There are the three main asset Categories of CAVs, RSU and intelligent transport systems (ITS) central station as discussed earlier. The architecture of the system incorporating security is proposed later in this section.

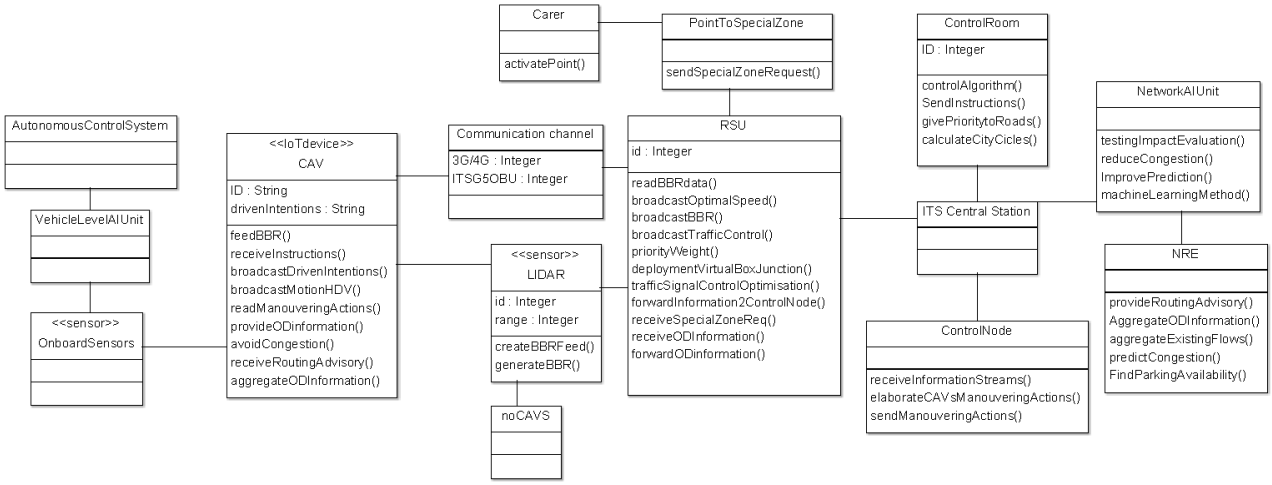

Figure 1: Original FLOURISH data architecture

CAVs hold onboard sensors, the vehicle level AI unit and the autonomous control, besides some attributes such as an ID and driving intensions (represented as a software module). The operations that the CAVs perform include: *feedBBR*, *receiveInstructions*, *broadcastDrivenIntensions*, *readManoeuvringActions* and *provideODinformation*. Each one of these instructions corresponds to one functionality described in the scenarios, e.g. the *broadcastDrivenIntensions* operation correspond to the use case of maneuvering collaboration where CAVs must broadcast their driving intentions to other CAVs in order for them to correctly react to the new movements and even predict new driving intensions. The communication channel between CAVs and RSU may be achieved through two ways, the first one by 3G/4G connectivity and the second one by the ITSG5OBU standard. These two ways allow the CAVs to send and receive data from the RSU.

The RSU operations include: *readBBRdata*, *broadcastOptimalSpeed*, *broadcastBBR*, *broadcastTrafficControl*, *priorityWeight*, *deploymentVirtualBoxJunction* and *trafficSignalControlOptimisation*. The principal RSU functionality is to receive information from the processing nodes and forward data to the CAVs. The operations correspond to the different kinds of data that the RSU must forward. The network AI unit is modelled with the **NetworkAIunit** class therefore, its operations correspond to the network AI unit behavior. The control room is modelled with the **ControlRoom** and its operations (*controlAlgorithm*, *SendInstructions*, *givePriorityRoads* and

calculateCityCicles) are focused on give priority to certain roads and send instructions to the RSU. The Carer is the person who activates the special zone through the point dedicated to activating it. This asset is modelled by the **Carer** class and it includes one operation *activatePoint*. LIDAR is the asset which monitors the CAVs and HDV, it obtains and creates data about the movements. The operations defined for the LIDAR class are *createBBRFeed* and *generateBBR*.

In order to facilitate the threat identification, an intuitive and graphic notation is needed, Attack trees diagrams are proposed to that effect. Attack trees are an orderly and sequential way of describing the sub-attacks to violate a system, they are a useful tool to conceptualize and visualize the possible attacks, allowing the analyst to create attacker profiles, in order to make decisions about the possible mechanisms and security controls needed to protect the system.

We use 'SecureItree' for the threat surface analysis of the Flourish project. SecureItree is an attack tree modelling tool built by the Canadian company Amenaza (Spanish for threat) (n.d.). In this tool, the root node represents the end objective and the children nodes the different sub-attacks in order to accomplish the overarching goal. Nodes can be AND operators, OR operators, or a LEAF. The AND operator means that all of the children nodes are needed to accomplish the parent node. On the other hand, the OR operator means that any of the children nodes satisfy the parent node.

Subsequently the security of the system can be analyzed through the application of various scenarios. Due to page limit constraints, only one such scenario is explored in this paper, as an illustration of the application of our approach. One particular concern is the threat associated with the 'Spoofing of BBR data' (fig. 2). This is a situation where an attacker is able to falsify data, in this case the BBR data which is generated by the LIDAR sensor (BBR data falsification). We will use attack trees to model the attack and, in this case, the resulting structure involves the next sub attacks:

- Tamper with the on-board sensors: The on-board sensors are manipulated to change the data readings in order to create false information and as a consequence change the CAV's BBR data. The countermeasure is tamper-proof hardware and software for the on-board sensors.
- Impersonate the CAV sensor node: Another way to change the BBR data from CAV is a man-in-the-middle (MITM) attack. If an attacker is able to impersonate the CAV sensor node, then it can receive and change the on-board sensors data. The countermeasure is the authentication of the on-board sensors.
- Create a fake RSU: This attack is about creating a false RSU in order to perform a MITM attack, in this way the LIDAR would not be able to identify the false RSU, and it would share the BBR data. The countermeasure proposed against this attack is authentication of the RSU.
- Create a fake processing node: The MITM attack here is deployed between the LIDAR and some of the processing nodes. The countermeasure proposed to mitigate this threat is the authentication of the LIDAR and a trusted processing node.
- Tampering with the LIDAR: this attack is about physical tamper with the LIDAR, in order to change the data which is about to send. Countermeasures involves the use of tamper protection.

Once the attack trees have been defined and countermeasures have been identified, it is time to specify where the latter have to be placed. The IoTsecM profile includes extensions to the use cases metaclasses. The first step is to identify which system actor carries out the security countermeasures identified. Therefore, according to the scenarios proposed by the FLOURISH team the use case diagrams for each scenario adding the security countermeasures are drawn.

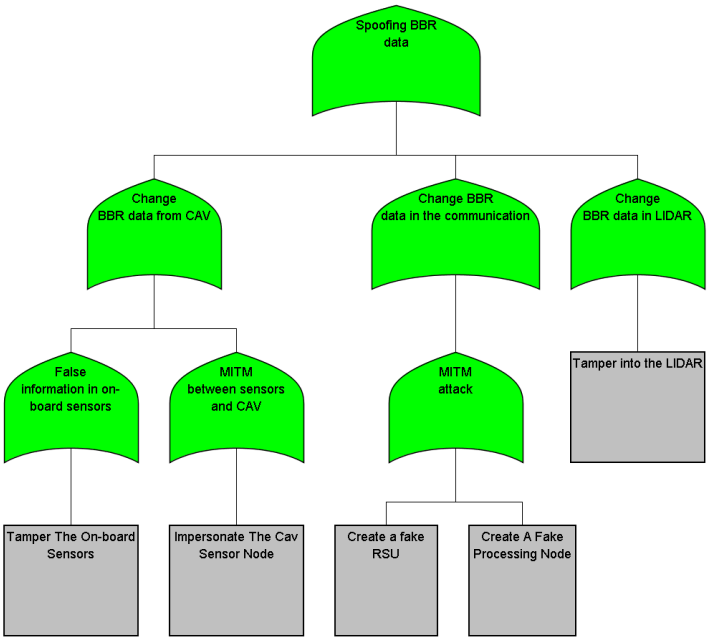

Figure 2: Spoofing BBR data attack tree

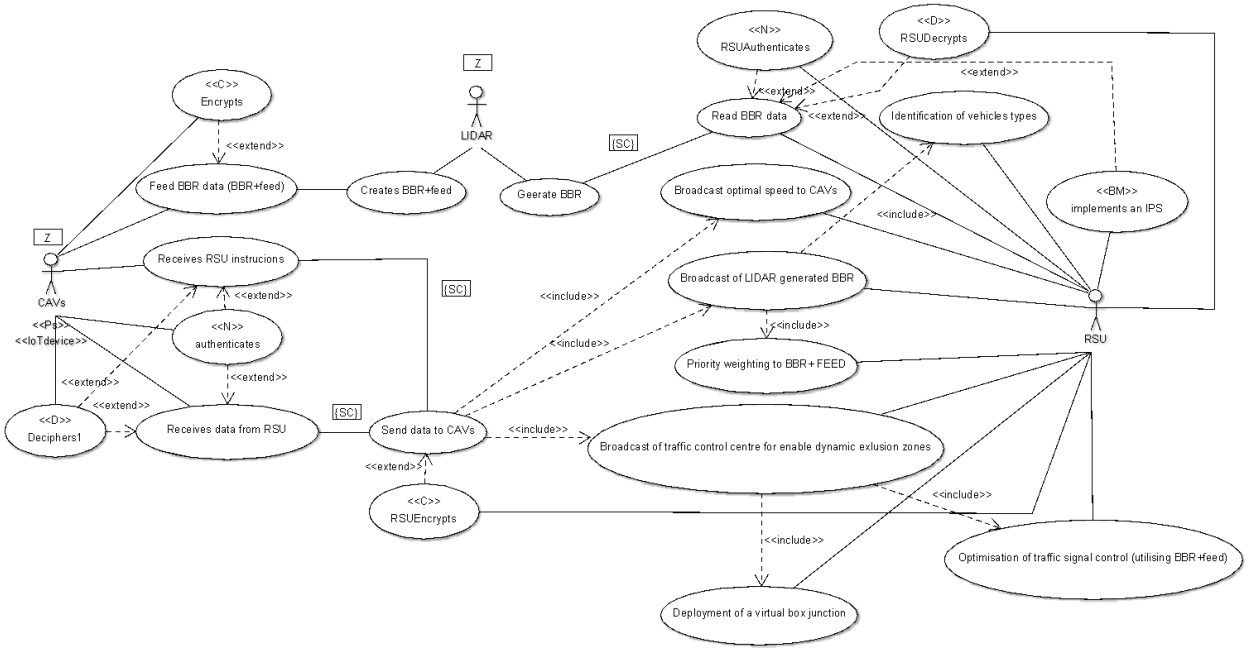

Figure 3: LIDAR scenario use-case diagram

The scenario described here is about the LIDAR and its interactions with the CAV and RSU. It is worthy of noting that the use-case diagram in Fig. 3. and the accompanying text documentation are in

situations where they relate to the identified countermeasure. The use case relating to the countermeasures are presented in tables III.a and III.b. The tables comprise the following fields: Use case name, participating actor, entry condition, flow of events and exit condition.

Use case name: <<N>> authenticates
Participating actor: CAV
Entry condition: An entry package is sent from RSU
Events flow: The package is received. The CAVs actor runs the authentication element. The <<N>> element obtains the RSU credentials from the package. The <<N>> stereotype instance creates complementary information from de credentials. The <<N>> stereotype instance runs the authentication function. The <<N>> creates the assertion {True, False}. This use case extends the Receives RSU instructions use case and Receives data from RSU use case
Exit condition: The CAVs authenticate the package received

Table III.a: <<N>> authenticates use case for CAV

Use case name: <<N>> RSU Authenticates
Participating actor: RSU
Entry condition: Receive data from the LIDAR
Events flow: The package is received. The RSU actor runs the authentication element. The <<N>> element obtains the LIDAR credentials from the package. The <<N>> stereotype instance create complementary information from de credentials The <<N>> stereotype instance runs the authentication function. The <<N>> creates the assertion {True, False}. This use case extends the Read BBR data instructions use case.
Exit condition: The RSU authenticates the package received.

Table III.b: <<N>> use case for CAV

Aside of the security use cases defined before, there are other constraints displayed on the use case diagram which are placed there to integrate more security concerns within the architecture. Also, there are two links identified as secure communications constraints. The links are the 'Receives data from RSU' and 'send data to CAVs' where the communication from RSU to the CAVs is established. The other Secure Communication ({SC}) constraint appears in the link between the LIDAR and the RSU. The LIDAR needs to be an authorized actor in order to be able to send data to the RSU. The RSU also needs to be authenticated, thus the "N" text box is placed over its head. The analysis of the countermeasures identified allows to specify the location where the security mechanism should be allocated. The use-case diagrams were very useful for security requirements conceptualization, as the IoTsecM extensions within the use case diagrams could represent each security countermeasure identified in the attack trees clearly.

The next step was to propose a whole-system architecture, including both functional and the non-functional elements in a class diagram. This helps to attend all the issues concerning to the interconnections between the assets, the identification of their operations and any relationships between the security mechanisms. The IoTsecM profile includes extension for classes, components and devices metaclasses, which assist the designing of the system architecture. In Fig. 4. the system architecture regarding the security elements is presented. The objective of the IoTsecM profile is to allow the designers to build, model and depict the security mechanisms together with the functional elements. As it can be seen in Fig. 4, security countermeasures identified are included in the system architecture. Here the CAV requires tamper protection and secure storage, besides requiring a pseudonym.

As shown in the use cases, the CAV authenticates and monitors the entry data and the network, hence, the <<N>> stereotype is instantiated. Likewise, the RSU must contain the security countermeasures found. Therefore, the stereotype instantiated and associated to the RSU is <<N>>; besides as well as the CAV, the PPKI infrastructure is supported by the RSU, hence a tamper protection is placed as a requirement. IoTsecM allowed us to capture security requirements by applying the stereotypes described. Once threat analysis was concluded, the countermeasures were identified and integrated with the functional requirements through use case and class diagrams as

discussed here. The UML notation provided a better understanding of where the security countermeasures needed to be placed, which actor is associated to them and how they are related to other system assets.

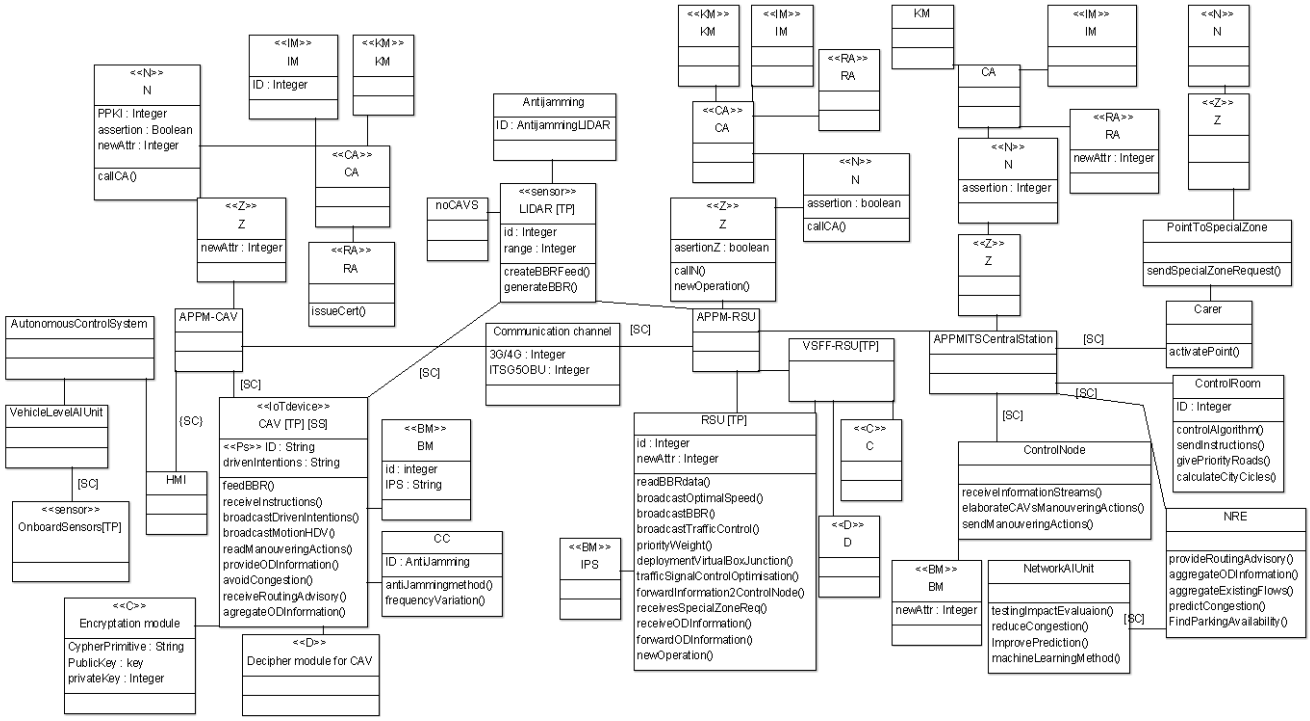

Figure 4: Re-engineered Flourish system architecture applying the IoTsecM profile

4 Conclusions and Further Work

The IoTsecM approach allowed to capture the security requirements by applying the UML stereotypes described earlier. Once threat analysis was performed, the countermeasures were identified and included in the design along with the functional requirements in use case and class diagrams. The UML notation provided a better understanding of where the security countermeasures needed to be placed, which actor is associated to them and how they are related to other system assets.

In the future, this architectural view may be extended with behavioral diagrams where the use cases and objects actions are included, in order to understand the processes followed by them, besides their interaction.

5 Acknowledgments

This work was supported by InnovateUK under Grant 102582, the FLOURISH Project. David Robles-Ramirez was supported by a CONACyT scholarship on a postgraduate placement at the University of Bristol, UK.

References

Amenaza website: http://www.amenaza.com/index.php, online (n.d.).

Apvrille L., Roudier Y. (2016) SysML-Sec Attack Graphs: Compact Representations for Complex Attacks. In: Mauw S., Kordy B., Jajodia S. (eds) Graphical Models for Security. GraMSec 2015. Lecture Notes in Computer Science, vol 9390. Springer, Cham

Baig, Z. et al. (2017). Future challenges for smart cities: Cyber-security and digital forensics. Digital Investigation, 22, 3-13.

Conti, M., Dehghantanha, A., Franke, K. & Watson, S., (2018). Internet of Things security and forensics: Challenges and opportunities, Future Gen. Computer Systems, 78, 544-546.

FLOURISH Insurance & Legal Report (2019). Annual Project Publication, available on-line: http://www.flourishmobility.com/publications

FLOURISH website: http://www.flourishmobility.com/, online (n.d.).

Jürjens, J., Schreck, J., & Yu, Y. (2008). Automated analysis of permission-based security using UMLsec. In International Conference on Fundamental Approaches to Software Engineering (pp. 292-295). Springer, Berlin, Heidelberg.

Khan, M. A., Iqbal, M. M., Ubaid, F. et al. (2016). Scalable and secure network storage in cloud computing. Intl Journal of Computer Science and Information Security, 14(4), 545.

Mansor, H., Markantonakis, K., Akram, R. N., Mayes, K., & Gurulian, I. (2016). Log your car: The non-invasive vehicle forensics. In IEEE Trustcom/BigDataSE SPA, pp. 974-982.

McCarthy, J., Bradburn, J., Williams, D., Piechocki, R., & Hermans, K. (2016). Connected & autonomous vehicles: introducing the future of mobility. Atkins.

Oates, R., Thom, F., & Herries, G. (2013). Security-aware, model-based systems engineering with SysML. In Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research (pp. 78-87).

Porter, M. E., & Heppelmann, J. E. (2014). How smart, connected products are transforming competition. Harvard Business Review, 92(11), 64-88.

Rigazzi, G., Tassi, A., Piechocki, R. J., Tryfonas, T., & Nix, A. (2017). Optimized certificate revocation list distribution for secure V2X communications. In 86th IEEE Vehicular Technology Conference (VTC-Fall).

Robles-Ramirez, D.A., Escamilla-Ambrosio, P.J. & Tryfonas, T. (2017). IoTsec: UML Extension for Internet of Things Systems Security Modelling. Intl Conf on Mechatronics, Elec and Automotive Engineering (ICMEAE), pp. 151-156.

Samie, F., Bauer, L., & Henkel, J. (2016). IoT technologies for embedded computing: A survey. Proceedings of the 11th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (p. 8).

Shostack, A. (2014) Threat modeling: Designing for Security. John Wiley & Sons.

Unis, M. et al. (2013). Internet of Things Architecture - A Final architectural reference model for the IoT v3. University of Surrey report, no. 257521.

Woo, S., Jo, H. J., & Lee, D. H. (2015). A practical wireless attack on the connected car and security protocol for in-vehicle CAN. IEEE Trans. Intel. Transp. Systems, 16(2), 993-1006.